DOI: 10.1109/AHS.2017.8046353

https://doi.org/10.24355/dbbs.084-202105041458-0

SEU Fault Classification by Fault Injection for an FPGA in the Space Instrument SOPHI

Holger Michel*, Hipólito Guzmán-Miranda†, Alexander Dörflinger*,
Harald Michalik*, Miguel Aguirre Echanove†

* Institute of Computer and Network Engineering
Technische Universität Braunschweig
Braunschweig, Germany
michel@ida.ing.tu-bs.de

† Universidad de Sevilla
Escuela Superior de Ingenieros
Grupo de Ingeniería Electrónica
Sevilla, Spain
hguzman@us.es

***Abstract*—Fault injection through partial dynamic reconfiguration can simulate upsets in configuration memory of SRAM-based FPGAs. FT-UNSHADES 2 is an automated set-up, which runs multiple fault injection campaigns in batch mode, while automatically applying stimuli and comparing output vectors. This work presents the results of fault injection runs of an FPGA design intended for the data processing unit (DPU) of the Solar Orbiter Polarimetric and Helioseismic Imager (SoPHI) instrument on solar orbiter. In this DPU SRAM FPGAs are connected to a processor through a radiation hardened antifuse FPGA. This antifuse FPGA houses the configuration and data interfaces to the SRAM FPGAs of the DPU. When radiation induced errors occur in the SRAM FPGA, the antifuse FPGA isolates these errors and recovers operation. The fault injection campaign gave insight on fault induced behavior on the interfaces of the SRAM FPGA, allowed to categorize them, and create statistics of the different categories. This paper describes the mechanisms of fault detection isolation and recovery in the SRAM/antifuse FPGA interfaces and tests them with the faulty output vectors from fault injection.**

I. Introduction

Scientific space probes and satellites consist of an space craft platform and various science instruments. The science instruments acquire measurement data and the space craft carries them to their orbit and performs communication with an earth station. Complex instruments require their own data processing unit (DPU) to control them. Those control tasks do not require a lot of processing power. Modern sensors are able to acquire large amounts of data, but the down link rate has not increased at the same rate, as it requires expensive ground stations. Therefore the Solar Orbiter Polarimetric and Helioseismic Imager (SoPHI) instrument is equipped with a more powerful DPU to compute and compress acquired data. In the SoPHI instrument, it is not the acquired image which is of scientific interest, but maps of magnetic field properties, that can be computed from acquired images. By not transmitting acquired images, but magnetic field parameters, the instrument can reduce data rate even more than by simple compression.

In order to be able to process these large amounts of data, the DPU requires more processing power than space qualified processors can offer. SRAM FPGAs are available in space qualified packages and can be programmed to perform data processing tasks faster than space grade processors. However, SRAM FPGAs are only radiation tolerant and do experience operational interrupts like SEUs for single memory cells or SEFIs for the entire device. Therefore the architecture of the DPU needs to be able to deal with this kind of events.

To test the design for operational interrupts caused by ionizing particle radiation on ground would require an expensive particle accelerator. Instead, susceptibility data from radiation tests by the manufacturer is combined with radiation data of the orbit to estimate upset rates. Upsets that alter the value of a memory cell in an SRAM FPGA can be grouped into three categories: Single event functional interrupts (SEFIs), upsets in user registers that change the value stored in a register of the design, and upsets that alter the value in the configuration memory. As the number of bits in configuration memory is the largest of these three categories, these upsets have the largest cross section. Errors in the configuration memory do alter the way the FPGA operates and thus the logic function it performs. Fault injection is the process of simulating such an upset by altering the value of such a memory cell [1]. Therefore the combination of radiation testing of the device, radiation data of the orbit, and fault injection allows to investigate and estimate the behavior of a system with an SRAM FPGA in orbit [2]. This paper presents such an estimation for the DPU of the SoPHI instrument.

II. Fault injection with FT-UNSHADES 2

FT-UNSHADES 2 [3] [4] is an automated FPGA-based fault injection platform. It consists of an FPGA motherboard and two custom daughterboards, see Fig. 2. A server running a user friendly web interface connects via USB 2.0 to the motherboard. The motherboard stores stimuli and configuration data and compares responses. It also connects to the daughterboards via PCI-Express. Each daughterboard features a service FPGA and a test FPGA. The service FPGA configures the test FPGA via SelectMap, applies input stimuli, and records output responses from the test FPGA. The design under test (DUT) can therefore fully occupy the test FPGA. FT-UNSHADES 2 can inject errors to user registers in the design or the configuration memory. User registers correspond to register

https://doi.org/10.24355/dbbs.084-202105041458-0

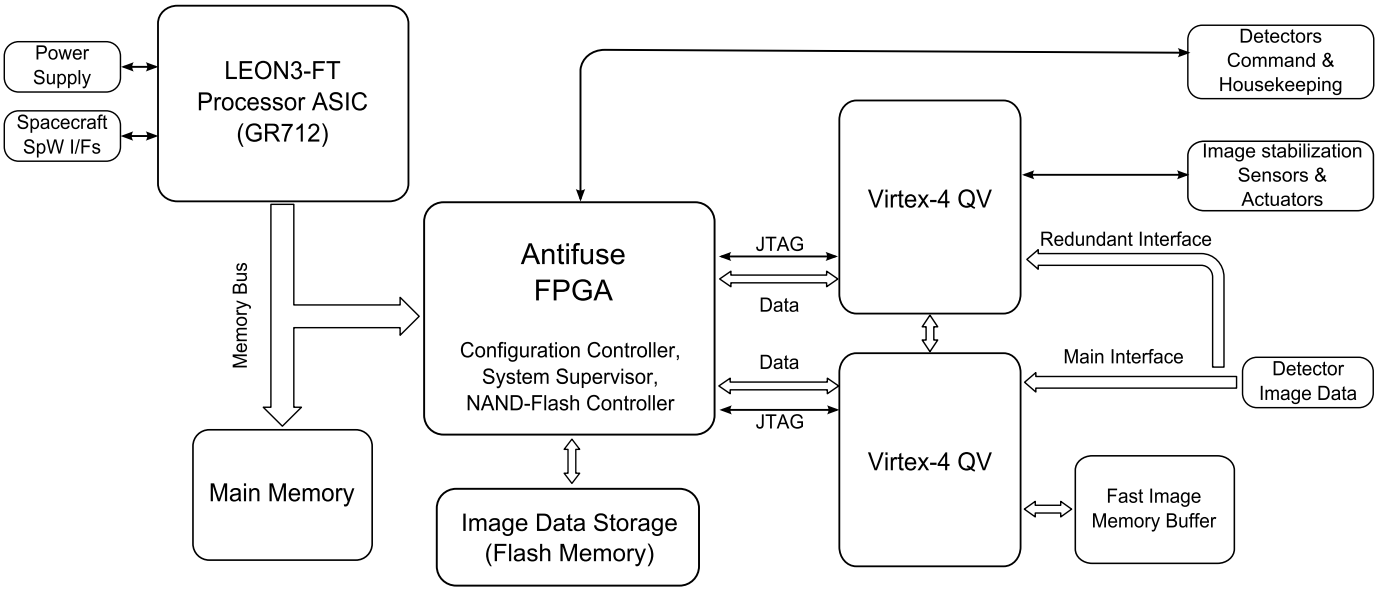

Fig. 1: Block diagram of the SoPHI DPU, with SRAM-based FPGAs (Xilinx Virtex-4 SX55), antifuse FPGA (RTAX), and processor (LEON3-FT).

values described in the VHDL design, they are therefore also present if the design is intended to be implemented in an ASIC. Therefore the user register mode can also be used to test ASIC designs. Injection into configuration registers only applies to SRAM FPGAs, so this mode is for testing the effects of upsets in configuration data. All FPGAs in the FT-UNSHADES 2 system are Xilinx Virtex-5 FPGAs. The test FPGA used in the test described in this paper is a Xilinx Virtex-5 FX70T. The custom daughterboards of FT-UNSHADES 2 can also be equipped with the LX50T variant.

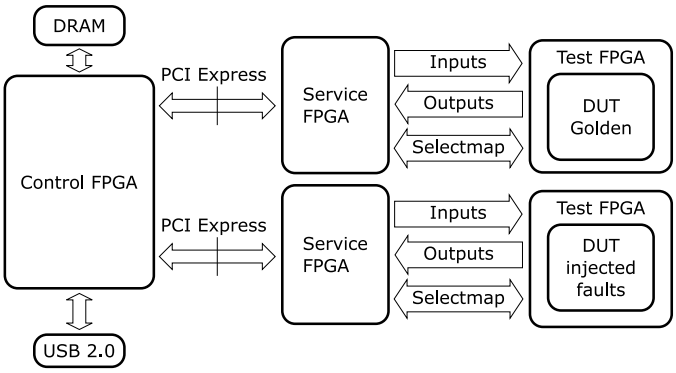

Fig. 2: Schematic of the FT-UNSHADES 2 setup; consisting of control FPGA, two custom FPGA cards containing a service FPGA and a test FPGA; each test FPGA hosts the design under test (DUT); via USB 2.0 this hardware connects to a host PC.

III. SoPHI instrument on Solar Orbiter

Solar Orbiter is a mission of ESA's Cosmic Vision 2015-2025 Program. It is currently in completion and its launch is scheduled for 2018. It will travel to close proximity of the sun and its observations are designed to improve understanding of solar and heliospheric physics [5].

The SoPHI instrument will acquire full solar disk and high resolution images at different polarizations and wavelengths. From this image data the DPU of solar orbiter will compute maps of the photospheric vector magnetic field, line of sight velocity, and continuum intensity in the visible wavelength range [6].

IV. Data Processing Unit of SoPHI

To meet all these processing requirements, the DPU of the SoPHI instrument features a GR712RC LEON3FT processor, a Microsemi RTAX FPGA, and two Xilinx Virtex-4 FPGAs [7]. Fig. 1 shows a simplified diagram of the interconnections between these integrated circuits. The GR712RC LEON3FT processor runs the instrument software. By attaching an RTAX FPGA to the IO-area of the processor's memory bus, software can read and write registers in that FPGA. The RTAX FPGA in turn connects to two Xilinx Virtex-4 FPGAs. These FPGAs are referred to as Reconfigurable FPGAs (RFPGAs) in this system.

Image data from the detector arrives at the Virtex-4 FPGAs. In default operation one Virtex-4 FPGA receives and buffers data and the other FPGA runs an image stabilization algorithm. In case the FPGA dedicated for image acquisition fails, the other FPGA can resume this task. However, this FPGA has no buffer memory resolution and therefore data rate would have to be reduced. The Virtex-4 FPGAs can be powered down, while the processor remains operational.

Fig. 3 is a diagram of design blocks in the antifuse FPGA. There is a data and configuration interface to each of the Virtex-4 FPGAs. For configuration the RTAX FPGA features a JTAG controller accessing the JTAG configuration port of

https://doi.org/10.24355/dbbs.084-202105041458-0

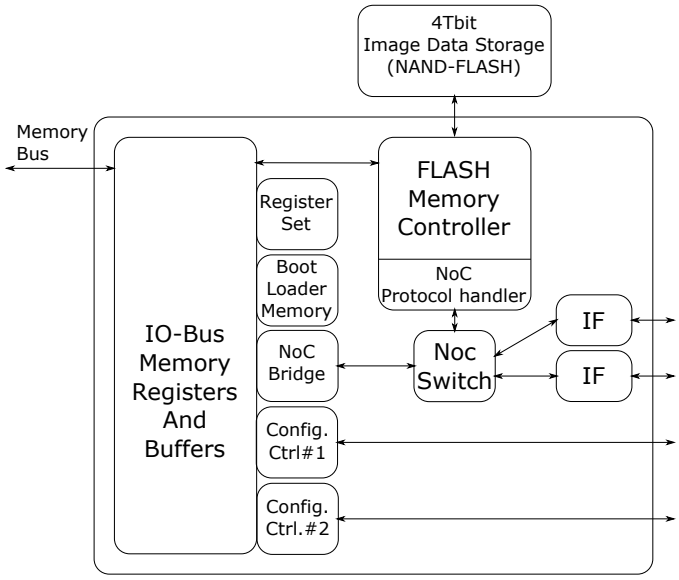

Fig. 3: Block diagram of the central control FPGA in SoPHI DPU.

each Virtex-4 FPGA. Due to lack of pins and to protect from errors resulting from upsets in the Virtex-4 FPGAs there is an interface block for data communication which performs serialization. In the following this block will be referred to as InterFPGA interface. The two InterFPGA interfaces attach to a wormhole routing switch. Wormhole routing implies that the switch consumes the first data word and passes it on to the connection indicated in that data word. The switch in turn connects to a flash memory controller for direct storage of acquired data and an IO-Bus bridge, to the memory bus of the processor. Via this bridge software running on the processor can send and receive data packets to and from the switch. The switch and attached nodes form a Network-on-Chip. This Network-on-Chip extends into the Virtex-4 FPGAs where it may feature another switch or directly terminate with a node.

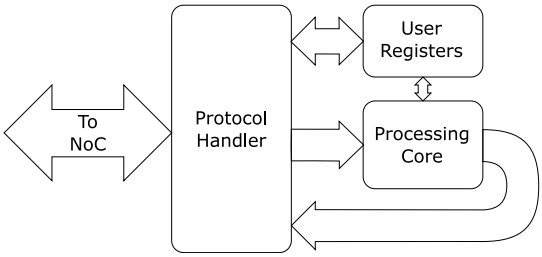

Fig. 4: SocWire protocol (SoCP) network node consisting of protocol handler, its registers, and the processing core.

Each node in the Network-on-Chip features a protocol handler, see Fig. 4. The protocol is derived from the ESA standard remote memory access protocol (RMAP), it is referred to as SocWire protocol (SoCP) [8]. It is adapted for on chip use and simplified, so that a hardware handler fits in every node. Each protocol handler has read and write registers to set parameters. The protocol handler processes the addressing header of packets and can pass bare process data to processing nodes. There are two types of packets, which the handler passes to the processing core. For *processing* packets the handler sends the reply of the processing core to the processor and for *stream* packets the reply is passed to another node. The address of the node to which data is passed can be set by registers in the protocol handler. In every switch in this Network-on-Chip the path to the microprocessor must always be on port number zero, thus the address of the processor is always known and does not need to be stored.

A. Design under Test (DUT)

The LEON3FT processor and antifuse FPGA feature triple modular redundancy in their registers and do not have a configuration memory that can be upset by ionizing particle radiation. However the two *RFPGAs* (Xilinx Virtex-4 FPGAs) are susceptible to upsets. To evaluate the effects of radiation upsets with fault injection the test design depicted in Fig. 5 was prepared. Inside the RFPGA is one side of a serialization interface (*interFPGA_if*), an SoCP protocol handler, and a finite state machine (FSM) with an attached internal memory block. This state machine is a simple example of a processing core. It allows to read and write the content of data packets to the memory block.

In simulation there is also the *interFPGA_if* that would normally be contained in the antifuse FPGA. It serializes packets sent to the DUT and deserializes the responses. This simplifies debugging and allows to attach a simple file reader to simulate the request that would normally be sent by the processor. The file used for simulation contains the following packets.

1) Writing zeros and ones to all registers.
2) Setting the handler register that is used as write address for the Block RAM.
3) A process packet to write 2 data words to Block RAM.
4) A stream packet to write 64 data words to Block RAM.
5) Writing to the register that holds the read address.
6) Register write to trigger reading and define read length.
7) Reading from read registers.
8) Testing with incomplete packages and parity errors.

The stimuli vectors were recorded from the waveforms of the simulation. Subsequently, FT-UNSHADES' service FPGAs played those stimuli vectors during fault injection.

B. Completeness of the test vectors

The test input vectors recorded from waveform simulations with the test bench are as good as the test bench. We used the code coverage metrics branch, expression, condition, and toggle coverage to ensure test set completeness. The ability of the input test vectors to allow an error caused by upsets to propagate to the outputs is beyond the scope of this paper, but an interesting field of further work.

https://doi.org/10.24355/dbbs.084-202105041458-0

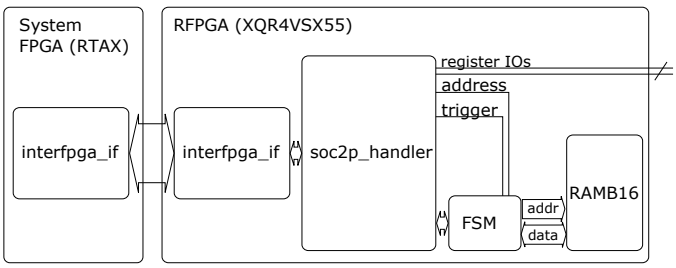

Fig. 5: Sketch of the tested design, with Inter FPGA interface, SoCP Protocol handler, FSM finite state machine, and Block RAM memory.

C. Upset rates estimated for the SOPHI instrument on Solar Orbiter

An important challenge when designing systems with integrated electronics for space applications are single event upsets [9]. The ionizing particle radiation that causes these upsets has been studied [10] and there is radiation data available for earth orbits. Solar Orbiter will operate on an elliptic orbit around the sun with a perihelion down to 0.28 AU [11]. The environment specification of solar orbiter recommends the use of radiation data from a geostationary orbit and scaling of the radiation components originating from the sun to account for closer proximity. This is computed by simply dividing the the surface of the sphere at the distance of the earth $r_E = 1AU$ by the surface of the sphere at the distance of the spacecraft $r_S = 0.28AU$.

$$\text{scaling factor} = \frac{4\pi(r_E)^2}{4\pi(r_S)^2} = \frac{1}{(\frac{r_S}{r_E})^2} = 12.7$$

For the estimation of upset rates 5 typical scenarios are used. Two of these scenarios do not need to be scaled as they are outside solar particle events.

1) Solar Maximum: The solar wind is at a maximum, so galactic cosmic rays are at a minimum.
2) Solar Minimum: The solar wind is at its minimum, so galactic cosmic rays are at their maximum.

The remaining scenarios need to be scaled, as in these cases radiation originates from the sun.

1) Solar Flare Worst Week: Average flux for the worst week of a solar particle event.
2) Solar Flare Worst Day: Average flux of the worst day of a solar particle event.
3) Solar Flare Worst 5 Minutes: Average flux of the worst 5 minutes of a solar particle event.

Radiation data was obtained from CREME [12], with the setting of $1g/cm^2$ of aluminum shielding. That radiation data was downloaded, modified with the scaling factor and uploaded again. [13] discusses this in detail.

Susceptibility data for individual elements of Xilinx Virtex-4 FPGAs is available from radiation testing, [14] and [15]. In CREME96 this data was combined with the radiation environment scaled for the orbit of solar orbiter, see [13] again for details. Table I and II present the expected error rates for the Virtex-4 FPGAs on the SoPHI DPU. The selected error categories are defined and explained in detail in [15].

TABLE I: Estimation of heavy ion induced SEUs and SEFIs per day and individual element.

SEUs per individual element				
	Config. Cell	BRAM	User FFs[1](1)	User FFs[1](0)
solar min	2.5e-7	7.1e-7	1.0e-6	3.9e-7
solar max	8.4e-8	2.6e-7	3.6e-7	1.2e-7
worst week	1.4e-3	9.6e-3	8.7e-3	7.9e-4
worst day	6.6e-3	4.8e-2	4.2e-2	3.6e-3
worst 5 min.	2.5e-2	1.7e-1	1.6e-1	1.3e-2
SEFIs[2] per device (entire RFPGA)				
	POR SEFI[3]	SMAP SEFI[4]	GSIG[5]	
solar min	1.2e-5	9.6e-6	3.9e-6	
solar max	4.0e-6	3.3e-6	1.4e-6	
worst week	0.26	0.2	0.095	
worst day	1.3	1.1	0.47	
worst 5 min.	4.7	4.3	1.8	

[1] Flip-Flop.
[2] Single Event Functional Interrupt.
[3] Power-On-Reset Single Event Functional Interrupt.
[4] SelectMap Single Event Functional Interrupt.
[5] Global Signal Single Event Functional Interrupt.

TABLE II: Estimation of proton induced SEUs and SEFIs per day and individual element.

SEUs per individual element				
	Config. Cell	BRAM	User FFs (1)	User FFs (0)
solar min	1.6e-8	1.7e-8	5.8e-8	1.7e-8
solar max	6.0e-9	6.1e-9	2.0e-8	6.1e-9
worst week	1.5e-4	2.6e-4	8.2e-4	2.3e-4
worst day	5.8e-4	1.0e-3	3.3e-3	9.2e-4
worst 5min	2.1e-3	3.8e-3	1.2e-2	3.3e-3
SEFIs per device (entire RFPGA)				
	POR SEFI	SMAP SEFI	GSIG	
solar min	8.3e-07	6.5e-7	8.1e-7	
solar max	3.0e-7	2.3e-7	3.0e-7	
worst week	8.6e-3	9.9e-3	7.7e-3	
worst day	3.4e-2	4.0e-2	3.0e-2	
worst 5 min	1.2e-1	1.5e-1	1.1e-1	

V. Comparison of Virtex-4 and Virtex-5 FPGAs

The SoPHI DPU uses two Xilinx Virtex-4 XQR4VSX55 FPGAs, but the FT-UNSHADES 2 boards are equipped with Xilinx Virtex-5 XC5VFX70T FPGAs. Firstly, data from radiation testing is available for all space grade devices, including the XQR4VSX55 FPGAs. Estimation of upset rates was there performed solely with Virtex-4 data, so physical size of elements and chip manufacturing is correctly accounted for. Fault injection is only used to investigate design behavior when configuration bits change their value. Thus, there is no difference in using commercial devices instead of flight parts for fault injection.

Virtex-5 devices are the direct successor to Virtex-4 devices. In both devices a configurable logic block (CLB) forms the smallest addressable block for implementation of logic functions. In a Virtex-4 such a CLB contains 4 slices and in a Virtex-5 a CLB contains 2 slices, but each Virtex-4 slice has only two 4-input look-up tables whereas a Virtex-5

https://doi.org/10.24355/dbbs.084-202105041458-0

slice has four 6-input look-up tables. In conclusion a Virtex-5 CLB can implement more complex logic functions in a single CLB. The synthesis tool takes the look-up table structure into consideration. Xilinx ISE allows to use a netlist synthesized for Virtex-4 on a Virtex-5 device. Table III lists slice usage for the interface test design used for fault injection.

TABLE III: LUT usage of Virtex-4 and Virtex-5 designs.

Design	*Device*	*# of slice LUTs*	*% of slice LUTs*
IF test	Virtex-5	763	1%
IF test	Virtex-4	1076	2%
IF test	Virtex-5 (synthesis Virtex-4)	914	2%

The number of LUTs used in a Virtex-4 design and a design synthesized for Virtex-4, but implemented on Virtex-5, are similar and differ significantly from a design synthesized and implemented on a Virtex-5. The similarity of the structures and these numbers indicate that it is viable to make estimations from usage of a Virtex-5 device in fault injection instead of a Virtex-4 device.

VI. Erroneous Outputs per Injected Fault

The first objective of the fault injection campaign is to establish how many of the 22,702,848 configuration bits in a Virtex XQR4VSX55 actually lead to a detectable error when altered through an upset. Even with the ability to inject many faults quickly of FT-UNSHADES 2 it is not possible to inject in every configuration bit at every possible clock cycle of the input stimuli. For Virtex-5 implementation the Xilinx Impact tool delivers a set of essential bits. The fault injection results listed in Table IV are obtained by 10,000 injections to those essential bits.

TABLE IV: Number of injections and recorded errors in the output vectors.

Design	*Synthesis*	*errors*	*injections*	*%*	*essential bits*
IF test BRAM	Virtex 5	5453	10000	55	163783
IF test BRAM	Virtex 4	5376	10000	54	178789
IF test dRAM	Virtex 5	5362	10000	54	331501
IF test dRAM	Virtex 4	5497	10000	55	1216979

Each injection was equivalent to flipping one random configuration bit at a random clock cycle and running it for all clock cycles of the input vectors. Errors were counted if there were one or more errors in the output vectors of a run. In all four tests listed in Table IV about 54% of the essential bits showed detectable errors at the outputs. As the Virtex-4 synthesis which uses Block RAMs is the closest to the design that will be used in the DPU, this results in an estimation of 54% of the $178,789$ essential bits actually resulting in errors, when flipped. So the number of actually sensitive bits is about $100,000$ for this design.

The effective error rates listed in Table V show that the design has few errors under normal condition (solar min and max). Solar Orbiter's orbit is very elliptical, so it is unlikely that flare conditions at closest approach coincide. However if a design shall function in these conditions, it needs to be protected by triple modular redundancy.

TABLE V: Estimation of errors resulting from SEUs in configuration memory per day, assuming $100,000$ actually sensitive bits and susceptibility per individual element as given by Table II.

conditions	*errors per day*	*errors per 5 min*
solar min	0.0016	
solar max	0.0060	
worst week	15	
worst day	58	
worst 5min		0.73

VII. Fault classification

FT-UNSHADES does not only detect errors in outputs, it does also record XOR vectors of the output of the golden DUT and the injected DUT along with the clock cycles at which the differences occurred. A script can reconstruct the faulty outputs as given by the DUT.

A. Characterization by affected outputs

The most obvious way to characterize errors in outputs due to fault injection is to categorize them by the output bits affected. A matlab script processed the 10,000 fault injection runs for all four design variants. Table VI lists the results from processing these fault injections runs. The injection runs have been categorized in five categories:

1) In the InterFPGA interface there is a READY signal that indicates that the receiver is able to process more data. If it is low, this signal stalls the transmitter in the other FPGA. The READY signal output of the Virtex-4 FPGA is an input to the antifuse FPGA. Runs that only resulted in errors in this READY make up this first category.
2) This category contains all injection runs with errors in signals of the InterFPGA interface other than the READY signal. Therefore this category reflects injections that lead to errors in the data packets.
3) As shown in Fig. 5 the protocol handler has registers directly connected to outputs. In the DPU these signals are used to as non-essential external control lines. This category includes injections, which only resulted in errors in these control lines.
4) This category contains injection runs that have error in the READY signal as well as other signals of the InterFPGA interface.
5) The final category contains all other cases, i.e. there are error in the register values and the InterFPGA interface outputs.
6) No error.

The percentages for the different categories are about identical. Slight variations for the different designs are no surprise, because the 10,000 injection locations for each design were randomly selected. With about 30% the majority of injection results fall into the first category. So they only stall the

https://doi.org/10.24355/dbbs.084-202105041458-0

TABLE VI: Number of injection runs that resulted in errors of the categories defined above.

Design	*Category*					
	1	*2*	*3*	*4*	*5*	*6*
V5 BRAM	2966	427	15	2009	36	4547
V5 dRAM	3009	334	15	1966	38	4638
V4 BRAM	3011	339	22	1964	40	4624
V4 dRAM	2962	471	12	2012	40	4503

reception of further data packets from the antifuse FPGA. This error condition is easily detectable by a timeout in the antifuse FPGA. The second category with errors in the packet data only accounts for 3% to 5% of the injection runs. It is therefore relatively rare. Category number 4 with errors in the READY signal as well as the other outputs of the InterFPGA interface, is more common with about 20%. Register outputs are only affected in very few cases, as categories 3 and 5 are below 1%.

B. Characterization by feeding outputs back to simulation

To investigate injection results the outputs of the fault injection campaign were fed back to the simulation test bench. This was accomplished by replacing the UUT by a file reader that reads the outputs recorded from the fault injection campaign, see Fig. 6. The receiver part of the InterFPGA interface performed deserialization and subsequent assertions reported errors in the data packets.

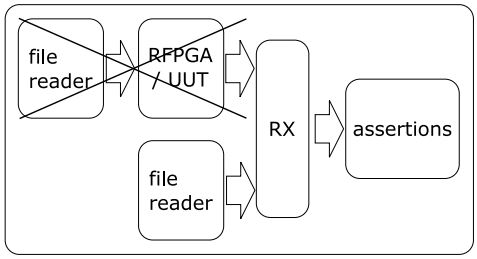

Fig. 6: Simulation without unit under test to evalutate reactions of interFPGA interface to fault injection outputs.

It is very straightforward to predict the reaction of the receiver on a stuck READY signal. It will simply stall and timeout. Therefore injection results of category 1 were not fed back to the simulation. Table VII lists observations from 10 injection campaigns that had errors in the data packets only, i.e. category 2.

The InterFPGA interface has 4 parallel data lines. The Network-on-Chip in both FPGAs operates with 16 parallel signals and thus 16 bit data words. The Receiver therefore assembles 4 times 4 bits back to a single word. There is an extra signal line on that interface that indicates control characters. When this line is high the data lines indicate one of three possible control character: 1) *Null* (there is no valid data in this clock cycle), 2) *End of Packet* (the data packet ends with this character), and 3) *Erroneous End of Packet* (the data packet ends with this character and the packet is marked, because an error occurred during its processing). With the end of packet markers it is possible to detect if the number of transmissions does not match up to 16 bit. The receiver performs this and marks those packets with an error labeled 'unmatched'. All but one injection result showed this error, which makes it simple to detect. Further observations were errors in stream packet data, hardware identifier, protocol instruction, and address.

TABLE VII: Observations in simulation with output data from injections that resulted in errors in the SoCP packets only.

Number	*Unmatched*	*Further observation*
1	Yes	Error in stream packet data
2	Yes	None
3	Yes	None
4	Yes	Error in Hardware ID
5	No	Error in SoCP instruction
6	Yes	Error in SoCP instruction
7	Yes	None
8	Yes	None
9	Yes	Error in SoCP instruction
10	Yes	Error in address

Injection results that only had errors in the register outputs were not investigated by these simulations, because the effect and results are obviously only faulty values on those outputs. Table VII lists the results from simulation of 5 cases with errors in all interFPGA signals including the READY signal. All of these cases led to a restart of the interFPGA are thus easily detectable.

TABLE VIII: Observations in simulation with output data from injections that resulted in errors in the SoCP packets and the READY signal.

Number	*Unmatched*	*Further observation*
1	Yes	link restart
2	No	link restart
3	No	link restart
4	No	link restart
5	No	link restart

Simulations with the last category, i.e. errors in the outputs that belong to the interFPGA interface and the registers, also resulted in easily detectable link restarts.

TABLE IX: Observations in simulation with output data from injections that resulted in errors in different output signals.

Number	*Unmatched*	*Further observation*
1	No	link restart
2	Yes	link restart
3	Yes	link restart; error in stream data
4	Yes	link restart; error in stream data
5	Yes	link restart; error in stream data

VIII. Conclusion

With FT-UNSHADES 2 the design is easily prepared for fault injection. Input vectors are simply recorded from the simulation test bench. FT-UNSHADES 2 automatically compares output vectors to a golden copy. The webinterface is easy to use and allows remote access. With this ease of

https://doi.org/10.24355/dbbs.084-202105041458-0

design preparation and the ability to perform many injection campaigns quickly in batch mode FT-UNSHADES 2 has made this study possible.

The fault injection campaign showed that only about 54% of flips in the essential bits lead to an observable misbehavior. Fault injection also showed the ways the design behaves in case of bit flips in configuration data. In about 55% of the cases the FPGA refuses to accept further data, which is easily detected by a timeout. Further measures like ensuring that the number of bits in a word match and protection on the data link were able to detect many cases of bit flips. There were instances in which the errors were solely in the data packets. So the usage and application needs to have the capability to tolerate the occurrence of errors in the application data, but the set-up ensures there is no case that results in a crashed DPU.

With the Xilinx Zynq FPGAs in space qualified package becoming available soon, an interesting aisle of further work is to update the injection platform to these devices. As newer FPGA generations use a standard AXI interface, it would be interesting to investigate AXI interfaces in a similar fashion.

ACKNOWLEDGMENT

This work is part of the DFG Research Group FOR 1800 “Controlling Concurrent Change”. Funding for the Institute of Computer and Network Engineering (IDA) was provided under grant number MI 1172/3-1.

REFERENCES

[1] P. Ostler, M. Caffrey, D. Gibelyou, P. Graham, K. Morgan, B. Pratt, H. Quinn, and M. Wirthlin, Sram fpga reliability analysis for harsh radiation environments, Nuclear Science, IEEE Transactions on, vol. 56, no. 6, pp. 35193526, Dec 2009

[2] S. Rezgui, R. Velazco, R. Ecoffet, S. Rodriguez and J. R. Mingo, Estimating error rates in processor-based architectures, in IEEE Transactions on Nuclear Science, vol. 48, no. 5, pp. 1680-1687, Oct 2001

[3] M. A. Aguirre, J. N. Tombs, F. Munoz, V. Baena, H. Guzman, J. Napoles, A. Torralba, A. Fernandez-Leon, F. Tortosa-Lopez, and D. Merodio, Selective protection analysis using a seu emulator: Testing protocol and case study over the leon2 processor, IEEE Transactions on Nuclear Science, vol. 54, no. 4, pp. 951-956, Aug. 2007

[4] H. Guzmán-Miranda, J. Nápoles, J. Mogollón, J. Barrientos, L. Sanz, and M. Aguirre, Ft-unshades2: A platform for early evaluation of asic and fpga dependability using partial reconfiguration, La Sociedad de Arquitectura y Tecnologa de Computadores, vol. 17, p. 26, 2012

[5] D. Müller, R. G. Marsden, O. C. St. Cyr, and H. R. Gilbert, Solar Orbiter, journal of Solar Physics, volume 285 number 1, pp 25-70, 2013

[6] A. Gandorfer, S. K. Solanki, J. Woch, V. Martínez Pillet, A. Álvarez Herrero, and T. Appourchaux, The Solar Orbiter Mission and its Polarimetric and Helioseismic Imager (SO/PHI), Journal of Physics: Conference Series, volume 271, pp 012086, 2011

[7] B. Fiethe, F. Bubenhagen, T. Lange, H. Michalik, H. Michel, and J. Woch and J. Hirzberger, Adaptive hardware by dynamic reconfiguration for the Solar Orbiter PHI instrument, NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp 31-37, 2012

[8] H. Michel, A. Belger, F. Bubenhagen, B. Fiethe, H. Michalik, W. Sullivan, A. Wishart, J. Ilstad The SoCWire Protocol (SoCP): A Flexible and Minimal Protocol for a Network-on-Chip NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2012

[9] E. Petersen, Single Event Effects in Aerospace IEEE Press Wiley, 2011.

[10] J. Barth, C. Dyer, and E. Stassinopoulos, Space, atmospheric, and terrestrial radiation environments, IEEE transactions on nuclear science, vol. 50, no. 3, pp. 466482, 2003.

[11] J. Sørensen Solar orbiter environmental specification ESA, Tech. Rep. TEC-EES-03-034/JS, 2010.

[12] https://creme.isde.vanderbilt.edu/CREME-MC/

[13] H. Michel, F. Bubenhagen, K. Grürmann, T. Lange B. Fiethe, and H. Michalik, Worst Case Error Rate Predictions and Mitigation Schemes for Virtex-4 FPGAs on Solar Orbiter, NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2013

[14] G. Swift, G. Allen, C. W. Tseng, C. Carmichael, G. Miller, and J. George, Static upset characteristics of the 90nm Virtex-4QV FPGAs, IEEE Radiation Effects Data Workshop, pp. 98105, 2008.

[15] G. Allen, G. Swift, and C. Carmichael, Virtex-4QV static seu characterization summary Jet Propulsion Laboratory Pasadena, California, Tech. Rep., 2008

https://doi.org/10.24355/dbbs.084-202105041458-0